Universidad de las Ciencias Informáticas

Facultad 3

Aplicación Android para la correcta visualización y funcionamiento del módulo control docente de AKADEMOS en dispositivos móviles.

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor:

Mauro Sergio Sotolongo Montes

Tutores:

Ing. José Alejandro García Calderón

Dr.C. Manuel Enrique Puebla Martínez

La Habana, septiembre de 2020

“Año 62 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaro que soyel único autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste se firma la presente a los _____ días del mes de _____ del año _____.

_____

Mauro Sergio Sotolongo Montes

_____

José Alejandro García Calderón

_____

Manuel Enrique Puebla Martínez

RESUMEN

La presente investigación aborda el del desarrollo de una aplicación Android (APK) del módulo control docente que se encuentra integrado al Sistema de Gestión Académica de Pregrado (AKADEMOS) de la Universidad de las Ciencias Informáticas. La APK desarrollada corrige los problemas de visualización existentes en el módulo, para su implementación se siguieron las pautas de diseño que propone “Mobile First” proporcionándole mayor accesibilidad, eficiencia y permitiendo una interacción más amigable para su uso desde dispositivos móviles. Para guiar el proceso de desarrollo se utilizó la metodología AUP-UCI la cual permitió la correcta definición por parte del cliente de las Historias de Usuario (HU), logrando así ver con claridad los requisitos funcionales necesarios para el modelado de la solución implementada. En el desarrollo de la aplicación se empleó Java y XML como lenguajes de programación y Android Studio como Entorno de Desarrollo Integrado (IDE en sus siglas en inglés). Lográndose de esta manera el cumplimiento de los requisitos definidos inicialmente y la validación de la solución mediante la completa satisfacción del cliente. Las pruebas realizadas arrojaron como resultados que el software implementado cumple con la calidad necesaria y satisface el objetivo de la investigación.

Palabras claves:AKADEMOS, control docente, registro de asistencia, registro de evaluaciones, visualización, Aplicación Android.

ABSTRACT

This research deals with the development of an Android application (APK) of the academic control module that is integrated into the Undergraduate Academic Management System (AKADEMOS) of the University of Computer Sciences. The developed APK corrects the existing visualization problems in the module. For its implementation, the design guidelines proposed by "Mobile First" were followed, providing greater accessibility, efficiency and allowing a friendlier interaction for its use from mobile devices. To guide the development process, the AUP-UCI methodology was used, which allowed the correct definition by the client of the User Stories (HU), thus achieving a clear view of the functional requirements necessary for the modelling of the implemented solution. In the development of the application, Java and XML were used as programming languages and Android Studio as an Integrated Development Environment (IDE). In this way, the fulfilment of the requirements initially defined and the validation of the solution through the complete satisfaction of the client were achieved. The tests carried out showed that the implemented software meets the necessary quality and satisfies the research objective.

Keywords:AKADEMOS, teacher control, attendance record, evaluation record, visualization, Android application.

ÍNDICE

Contenido

Introducción. 9

1.1 Principales conceptos. 14

1.2 Sistemas Homólogos. 14
1.2.1 Sistema de Gestión Universitaria – Akademos. 14
1.2.2 Esemtia. 15
1.2.3 Additio App. 15
1.2.4 iTeacherBook. 16

1.3 Tecnologías y herramientas. 17
1.3.1 Sistema Operativo Android. 17
1.3.2 Android Studio. (12) 19
1.2.3 JAVA. 20
1.3.4 Gradle. 20
1.3.5 XML. 20
1.3.6 pgAdmin3. 21
1.3.7 Pencil. 21
1.3.8 Postman. 21
1.3.9 Node.js 22
1.3.10 JSON 22
1.3.11 JWT 22

1.3 Metodología de desarrollo. 23
1.3.1 Variación de AUP para la UCI. 24

1.4 Conclusiones del Capítulo 1. 25
Capítulo 2: Análisis y diseño de la aplicación. 26

2.2 Especificación de Requisitos. 26
2.2.1 Requisitos Funcionales de la propuesta de solución. 27
2.2.2 Requisitos No Funcionales de la propuesta de solución. 28

2.3 Historias de Usuario. 29

2.4 Arquitectura de Desarrollo de Software Cliente – Servidor a tres capas. 31

2.5 Patrones. 32

2.5.1 Patrón Arquitectónico. 32

2.6 Patrón de Arquitectura del Sistema. 33

2.7 Patrones de Diseño del Sistema. 35
2.7.1 Patrones de asignación de responsabilidad (GRASP). 35
2.7.2 Patrones GOF (Gang of Four). 36

2.8 Modelo de datos. 37

2.9 Seguridad del Sistema. 38

2.10 Conclusiones del Capítulo 2. 38

ÍNDICE

Capítulo 3: Implementación y pruebas. **39**

3.1 Diagrama de Despliegue. **39**

3.2 Implementación de la solución. **40**

3.3 Pruebas de Software. **42**

3.3.1 Pruebas de Aceptación. **43**

3.3.2 Pruebas Unitarias **45**

3.3.3 Pruebas de Integración. **48**

3.4 Conclusiones del Capítulo 3. ..... **51**

Conclusiones Generales. **52**

Recomendaciones: **53**

Referencias **54**

ÍNDICE

Índice de Tablas

Tabla1. Sistemas Homólogos. 16
Tabla 2. Comparativa entre las características básicas o bases (home ground) ágiles y los rasgos. 23
Tabla 3. Requisitos Funcionales 27
Tabla 4. Historia de Usuario Listar Registros 30
Tabla 5 Ejemplo de Prueba de Aceptación 44
Tabla 6. Resultado de las pruebas realizadas por cada iteración 44
Tabla 7. Casos de prueba realizados para el camino base 47
Tabla 8. Caso de prueba de integración realizada al módulo Control Docente 51

ÍNDICE

Índice de Ilustraciones

Ilustración 1. Tendencia del tráfico de datos a nivel mundial. (1) 9
Ilustración 2. Problemas de visualización de AKADEMOS en dispositivos móviles. .. 10
Ilustración 3. Mapa de navegación de AKADEMOS. 15
Ilustración 4. Arquitectura de Android. (12) 19
Ilustración 5. Arquitectura Cliente-Servidor. 32
Ilustración 6. Relación Modelo-Vista-Controlador. 33
Ilustración 7. Estructura del módulo Control Docente siguiendo el patrón arquitectónico MVC. 34
Ilustración 8. Modelo de datos del módulo Control Docente. 37
Ilustración 9. Diagrama de Despliegue. 39
Ilustración 10 Menú Lateral. 40
Ilustración 11 Vista de la Pantalla Principal. 41
Ilustración 12 Vista del RF Registrar Asistencia. 42
Ilustración 13. Fragmento de código para buscar por nombre. 46
Ilustración 14. Representación del grafo de flujo del camino básico 47

INTRODUCCIÓN

Introducción.

"*Mobile first*" es un concepto utilizado en el diseño y la concepción de sitios web. Basándose en este, se crea una versión optimizada para dispositivos móviles y luego se amplía.(1)La estrategia "*Mobile first*" se enfoca principalmente en lograr aplicaciones más usables para usuarios que interactúan con estas desde un dispositivo móvil, ya que estadísticamente el mayor tráfico que se genera en Internet es desde un Smartphone o Tablet, y no desde computadoras de escritorio.

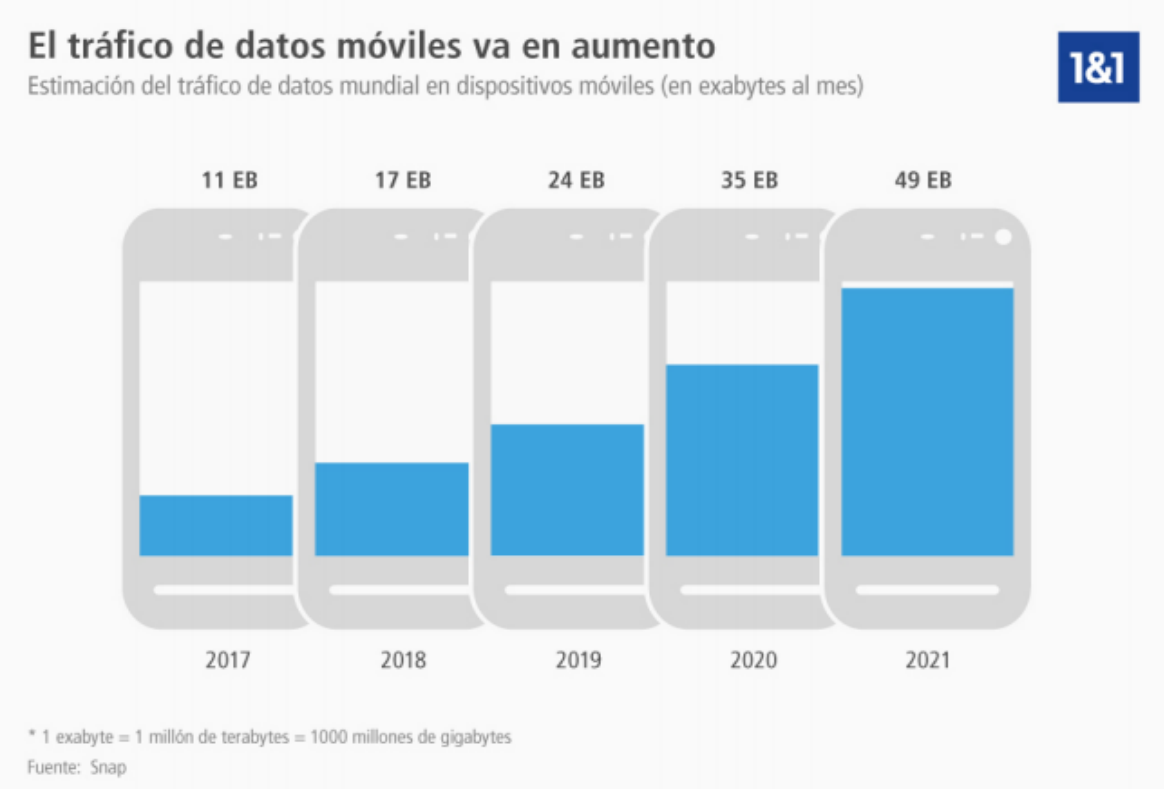

Ilustración 1.*Tendencia del tráfico de datos a nivel mundial.(1)*

Debido al alto tráfico de información generado desde dispositivos móviles, surge otra tendencia a nivel mundial conocida como "*Bringyourowndevice*", que consiste en que el personal de una empresa o institución puede hacer uso de dispositivos personales para realizar sus labores. Permitiendo de esta manera no estar atado a un espacio y lugar, tributando a una mayor productividad. (2)(3). La Universidad de las Ciencias Informáticas no queda fuera de este movimiento y como una solución al contratiempo de la conectividad ha provisto áreas wifi en puntos estratégicos y claves dentro del campus. Estas zonas de conectividad permiten el acceso a todos los servicios brindados desde la red universitaria.

INTRODUCCIÓN

Entre las soluciones desarrolladas enla Universidad de las Ciencias Informáticas (UCI), destaca una herramienta de apoyo personalizada denominada Sistema de Gestión Universitaria AKADEMOS, que constituye un sistema integrador que sirve de apoyo al registro y procesamiento de la información principal de los procesos sustantivos de la Universidad.(4), el cual gestiona las principales informaciones sobre el pregrado, ubicación laboral, postgrado e investigación. El sistema de pregrado cuenta con un módulo de control docente donde se realiza la gestión de los grupos docentes, registro de asistencia, registro de evaluación, registro de control a clases, cortes evaluativos y reportes.

Se ha podido verificar queel sistema AKADEMOS cuando se accede desde dispositivos móviles, no está desarrollado con un diseño completamente adaptativo, no se optimiza el uso del espacio en pantallas de pequeña y mediana resolución.Los componentes usados están enfocados a monitores de computadoras y en su mayoría se deforman en otro tipo de dispositivo, atentando así contra la usabilidad y productividad de los usuarios que requieren su uso desde sus propias terminales. A continuación, se muestra el comportamiento visual de algunas funcionalidades de AKADEMOS en dispositivos móviles. Como se evidencia en la primera interfaz, la funcionalidad de registro de asistencia visualizada desde una Tableta se hace prácticamente ilegible dificultando su uso. En la segunda interfaz, se muestra la pantalla principal del módulo de control docente inhabilitando la navegación en el menú de las funcionalidades del módulo. En la tercera interfaz, se muestra la funcionalidad de listar registros quedando en evidencia los errores de no aprovechamiento del espacio útil de la pantalla. De forma general queda demostrado que estos problemas limitan visualmente y funcionalmente el actual sistema de gestión académica en dispositivos móviles.

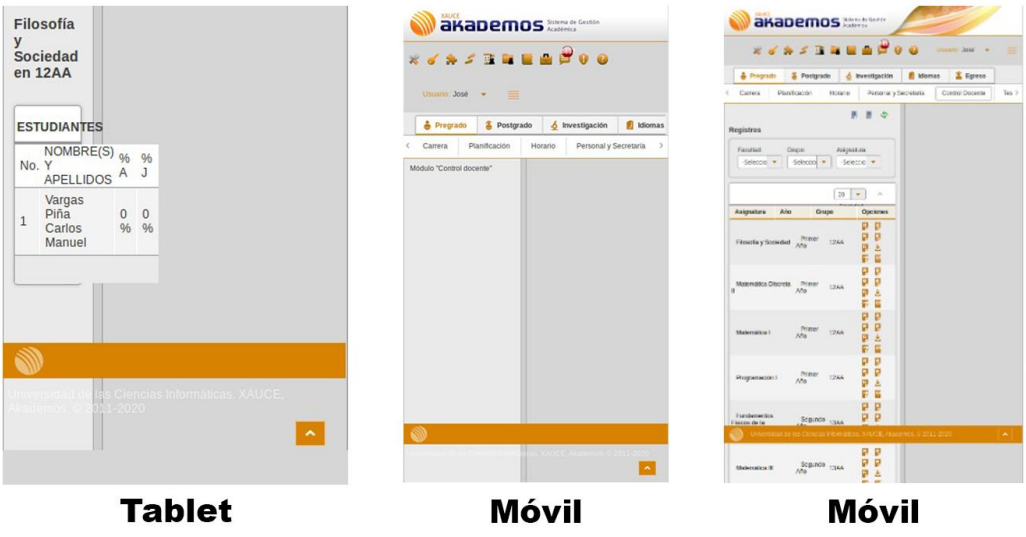

Ilustración 2. *Problemas de visualización de AKADEMOS en dispositivos móviles.*

INTRODUCCIÓN

Las tecnologías de desarrollo sobre las que está soportada actualmente AKADEMOS dificultan una modificación para lograr que el sistema cumpla con pautas para una correcta visualización en resoluciones pequeñas.La librería usada para el desarrollo de interfaz es jquery en su versión 1.9, la cual, actualmente ya no cuenta con soporte y al utilizar un alto número de componentes adaptados a las pautas de diseño de AKADEMOS se haceengorrosa una actualización de este y sus dependencias.

Por todo lo anteriormente planteado se deriva como **problema a resolver:** Los problemas de visualización en dispositivos móviles del módulo Control docente provocan disminución de las capacidades de interacción usuario-sistema.Para dar solución a este problema se tomará como **objeto de estudio:** Interfaces de usuario para dispositivos móviles. Teniendo como **campo de acción**: Interfaces de usuario para dispositivos móviles en el ámbito de la gestión académica.Asumiendo lo antes planteado, el **objetivo general** es**:** Desarrollar una aplicación Android para mejorar las capacidades de interacción usuario-sistema de AKADEMOS en los dispositivos móviles.

Para cumplir con el objetivo general expuesto se han trazado los siguientes **objetivos específicos:**

1. Construir el marco teórico de la investigación haciendo un análisis de los referentes teóricos, selección de la metodología, las herramientas y tecnologías a utilizar.
2. Realizar el análisis y diseño de la solución a implementar teniendo en cuenta las necesidades del cliente.
3. Ejecutar el proceso de pruebas y validación de requisitos,para garantizar la calidad de la solución implementada y la satisfacción del cliente.

La definición de los objetivos y el análisis del problema, guiaron a la obtención de la siguiente **idea a defender:** Si se desarrolla una aplicación Android que permita la correcta visualización del Módulo Control Docente de AKADEMOS, entonces mejorará la capacidad de interacción usuario-sistema.

Esperándose obtener como **posible resultado:** Una aplicación Android que mejore la capacidad de interacción usuario-sistema del Módulo Control Docente de AKADEMOS.

Con el propósito de cumplir con los objetivos específicos planteados se establecen las siguientes **tareas de la investigación**:

1. Delimitación del problema y caracterización del estado del arte de sistemas de gestión académica similares.
2. Definición de los métodos y técnicas de investigación a emplear.
3. Descripción del entorno y la metodología de desarrollo a emplear.

4. Definición de los requisitos funcionales y no funcionales según el proceso de desarrollo aplicado.
5. Modelar el sistema a través de Historias de Usuario.
6. Desarrollo del módulo.
7. Elaboración, ejecución y análisis de pruebas aplicables al módulo.

Los métodos de investigación utilizados para dar solución a la presente investigación son:

Métodos teóricos.

Con la misión de obtener conocimientos necesarios que hagan posible la materialización del objetivo general, se han utilizado diferentes tipos de métodos de investigación Teóricos y Empíricos, los cuales se describen a continuación:

Histórico-lógico: este método ha sido aplicado en la búsqueda realizada de la documentación sobre las aplicaciones móviles con tecnologías Android que permiten realizar registros docentes tanto de asistencias como de evaluaciones.

Analítico-sintético: este método ha sido aplicado en el análisis de la documentación identificada, para extraer los elementos que propicien la solución a la problemática planteada, así como la síntesis de los elementos necesarios para la selección de las tecnologías y metodologías adecuadas para el desarrollo de la aplicación Android propuesta.

Métodos empíricos.

Observación: La observación científica como método consiste en la percepción directa del objeto de investigación. Este método ha sido aplicado en la observación de lasinterfaces de las aplicaciones y el aprovechamiento del área útil de la pantalla. Es necesario lograr un diseño que permita visualizar correctamente las funcionalidades independientemente del tamaño de la pantalla. De esta forma, al diseñar una única vista que se adapte dinámicamente, permite reducir tiempo de producción.El presente trabajo de diploma está compuesto por tres capítulos, estructurados de la siguiente manera:

Capítulo 1. Fundamentación teórica de la aplicación Android.

En este capítulo se realiza la elaboración del marco teórico donde se exponen, valoran y comparan los conceptos asociados a la solución de la problemática y las diferentes soluciones existentes a nivel mundial. También se describen y caracterizan la metodología, herramientas y tecnologías a utilizar para el desarrollo de la aplicación Android.

Capítulo 2. Análisis y diseño de la aplicación Android.

Se especifica el Modelo Conceptual para identificar los principales conceptos del negocio. Se elabora los artefactos generados por la metodología seleccionada. Se modelan y detallan los diagramas que representan las funcionalidades del sistema, a partir de los patrones de diseño identificados.

Capítulo 3. Implementación, prueba y validación de la aplicación Android.

En el presente capítulo se lleva a cabo la representación del proceso de despliegue del sistema. Quedará documentado el proceso de pruebas llevado a cabo para garantizar la calidad del producto final. Se realizará una revisión final de los requisitos del sistema para verificar que se le dio cumplimiento a cada uno de ellos.

CAPÍTULO 1

Capítulo 1: Fundamentación teórica de la aplicación Android.

En el presente capítulo son abordados los principales conceptos utilizados en la investigación. Se describen las principales tecnologías, lenguajes de programación y herramientas definidas para el desarrollo de la solución, así como la metodología de desarrollo, los patrones a emplear y las métricas para la validación del diseño.

1.1 Principales conceptos.

La temática de esta investigación, relaciona varios conceptos que tributan directamente a esclarecer el tema de la correcta visualización de las aplicaciones en dispositivos móviles, así como a enriquecer las pautas a tener en cuenta para el desarrollo de interfaces que aprovechen de forma óptima el espacio útil en pantallas de resoluciones pequeñas. A continuación, se describirán los conceptos que se consideran más relevantes.

Descomponiendo un sistema en capas lógicas según su función, podemos ver a la **interfaz de usuario** como a la capa lógica encargada de dar soporte al diálogo con el usuario. Sus funciones son adquirir órdenes, información y comandos expresados por el usuario y presentar resultados, retroalimentación y cooperar para facilitar al usuario la realización de las tareas que pretende resolver el sistema.(5)

En términos de informática, estas funciones se enmarcan en el concepto de **usabilidad,**que se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicasde uso.(6)

Para lograr alcanzar un alto grado de usabilidad en un software es imprescindible hacer uso del **patrón de software**queayudan a construir la experiencia colectiva de Ingeniería de Software. Es una abstracción de “problema – solución”. Se ocupan de problemas recurrentes. Identifican y especifican abstracciones de niveles más altos que componentes o clases individuales. Proporcionan vocabulario y entendimiento común.(7)(8)

1.2 Sistemas Homólogos.

Existen varias aplicaciones para dispositivos móviles que permiten la gestión de registros docentes, a continuación, se describen algunas de las más calificadas:

1.2.1 Sistema de Gestión Universitaria– Akademos.

Según (Palmero y Remón, 2015) La UCI cuenta con un Sistema de Gestión Universitaria (SGU), desarrollado por el Centro de Informatización Universitaria, en el cual se gestiona toda la información referente a la formación de pregrado y postgrado. Cuenta con los siguientes módulos: Residencia, Pregrado, Egreso, Biblioteca, Ingreso, Laboratorios, Producción, Investigación, Cooperación y Postgrado. Los módulos anteriores se controlan

mediante un sistema de sesiones que garantizan el acceso eficiente a la información. El módulo Control Docente del subsistema de Pregrado dentro de sus principales funcionalidades tiene: Registro docente, Grupos docentes, control a clases. (9)

Ilustración 3. *Mapa de navegación de AKADEMOS.*

1.2.2Esemtia.

Del Grupo Edebé, es una plataforma integral que aborda todas las etapas educativas, desde Infantil a Formación Profesional. Ofrece las herramientas adecuadas para la gestión académica y administrativa del centro, así como para llevar a cabo el proceso de comunicación entre los miembros de la comunidad educativa. Sus productos EsemtiaSchool y Esemtia Enfant incorporan una función de justificación de faltas.(10)Es una herramienta privativa cuya adquisición constituiría un costo extra en USD a la institución.El uso de esta herramienta entra en contradicción con el lineamiento 108 del Partido Comunista de Cuba referente al desarrollo de soluciones propias para garantizar la soberanía tecnológica. (11)

1.2.3Additio App.

En web y para Android e iOS, funciona como un cuaderno de notas para que los docentes no sólo planifiquen el curso o lleven un seguimiento de las notas. Ayuda a controlar la asistencia al centro y es posible visualizar de manera instantánea los resúmenes de asistencia y elaborar informes. Se puede trabajar sin conexión a Internet sincronizando los datos entre dispositivos.(10)

1.2.4iTeacherBook.

Esta aplicación ha sido diseñada para controlar las tareas básicas del día a día del alumno de una forma sencilla e intuitiva. Con ella, se puede gestionar desde un único dispositivo la asistencia, las calificaciones, las fichas personales de cada estudiante e, incluso, generar informes sobre los progresos.(10)

El análisis realizado a las herramientas de registro académico y docente, permitió elaborar un resumen en forma de tabla teniendo en cuenta los siguientes parámetros:

- Web o Nativa: Referente a si la herramienta es una aplicación web o es nativa para sistema operativo Android.
- Interfaz adaptativa: Referente a si la herramienta se ajusta y optimiza el espacio en los diferentes tamaños de soluciones principalmente en dispositivos móviles.
- Tipo de licencia: Referente a si es libre de costo o es necesaria la compra de la misma.
- Procedencia: Referente a la procedencia del producto. Puede tomar valores Nacional y Extranjero.

Tabla1. Sistemas Homólogos.

Sistema / Característica	Web o Nativa	Interfaz Adaptativa	Tipo de Licencia	Procedencia
AKADEMOS	Web	No	Libre	Nacional
Esemtia	Ambos	Si	De Pago	Extranjera
Additio App	Nativa	Si	De Pago	Extranjera
iTeacherBook	Web	Si	De Pago	Extranjera

Se seleccionan estas características porque se consideran necesarias para dar cumplimiento a los requerimientos del software. Los sistemas estudiados fueron provechosos, pues sirvieron de referencias para la arquitectura de la información, proporcionando buenas prácticas en aprovechamiento del espacio útil de la pantalla de dispositivos móviles.

De los sistemas informáticos identificados con el objetivo de mitigar los problemas de visualización, aunqueen su mayoría aprovechanal máximo posible el espacio útil de la pantalla enlos dispositivos con resoluciones pequeñas, no se pudo seleccionar ninguno

para solucionar la problemática planteada porque no se adecuan a los procesos de la gestión académica en Cuba. La política de sustitución de importaciones y el difícil acceso a monedas libremente convertibles no hacen factible económicamente optar por aplicaciones con costos en su licencia. Todos estos elementos constituyen un factor importante para decidir apostar por el desarrollo de la aplicación con el módulo Control Docente para dispositivos móviles.

1.3 Tecnologías y herramientas.

En el presente epígrafe se describirán cada una de las tecnologías y herramientas utilizadas, especificando en cada caso con que versión se estará trabajando. Se decidió hacer uso de las mismas porque son las más recomendadas para el desarrollo de aplicaciones Android por las utilidades y características que estas aportan y se consideran adelantos para el desarrollador, tanto en la UCI como a nivel internacional.

1.3.1 Sistema Operativo Android.

Android es un sistema operativo que constituye una solución completa de software de código libre para teléfonos y dispositivos móviles. Es un paquete que engloba un sistema operativo, un "runtime"[1] de ejecución basado en Java, un conjunto de librerías de bajo y medio nivel y un conjunto inicial de aplicaciones destinadas al usuario final. Android se distribuye bajo la licencia libre permisiva (Apache) que permite la integración con soluciones de código propietario.(12)

Android surge como resultado de la Open Handset Alliance un consorcio de 48 empresas distribuidas por todo el mundo con intereses diversos de tecnología móvil y un compromiso de comercializar dispositivos móviles en este sistema operativo. El desarrollo viene avalado principalmente por Google (tras la compra de Android Inc. En 2005) y entre otras compañías, se encuentran compañías de software (Ebay, LivingImage), operadores (Telefónica, Vodafone, T-Mobile), fabricantes de móviles (Motorola, Samsumg, Acer, LG, HTC) o fabricantes de Hardware (nVidia, Intel o Texas Instruments).(12)

Arquitectura de Android.

Android presenta una arquitectura basada en 4 niveles como se muestra en la ilustración4 Arquitectura de Android:(12)

- Un**Kernel Linux** versión 3.14 que sirve como base de la pila de software y se encarga de las funciones más básicas del sistema: gestión de drivers, seguridad, comunicaciones, etc.

[1]Runtime: es el intervalo de tiempo en el que un programa se encuentra en ejecución en un sistema operativo.

- Una **capa de bibliotecas de bajo nivel en C y C++,** como SQLite para persistencia de datos; OpenGL ES para gestión de gráficos 3D, con aceleración 3D opcional y Webkit como navegador web embebido y motor de renderizado HTML.
- Un**framework para el desarrollo de aplicaciones**, dividido en subsistemas para gestión del sistema como el "Administrador de paquetes", el "Administrador de telefonía" (para la gestión del hardware del teléfono anfitrión) o el acceso a APIs sofisticadas de geolocalización o mensajería XMPP. Los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar el reúso de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mecanismo permite que los componentes sean reemplazados por el usuario. También incluye un sistema de vistas para manejar la interfaz de usuario de las aplicaciones, que incluyen la posibilidad de visualización de mapas o renderizado HTML directamente en la interfaz gráfica de la aplicación.
- **Aplicaciones:** Las aplicaciones base incluyen un teléfono, cliente de email, programa de envió SMS, calendario, mapas, navegador, contactos, que pueden a su vez ser usados por otras aplicaciones.

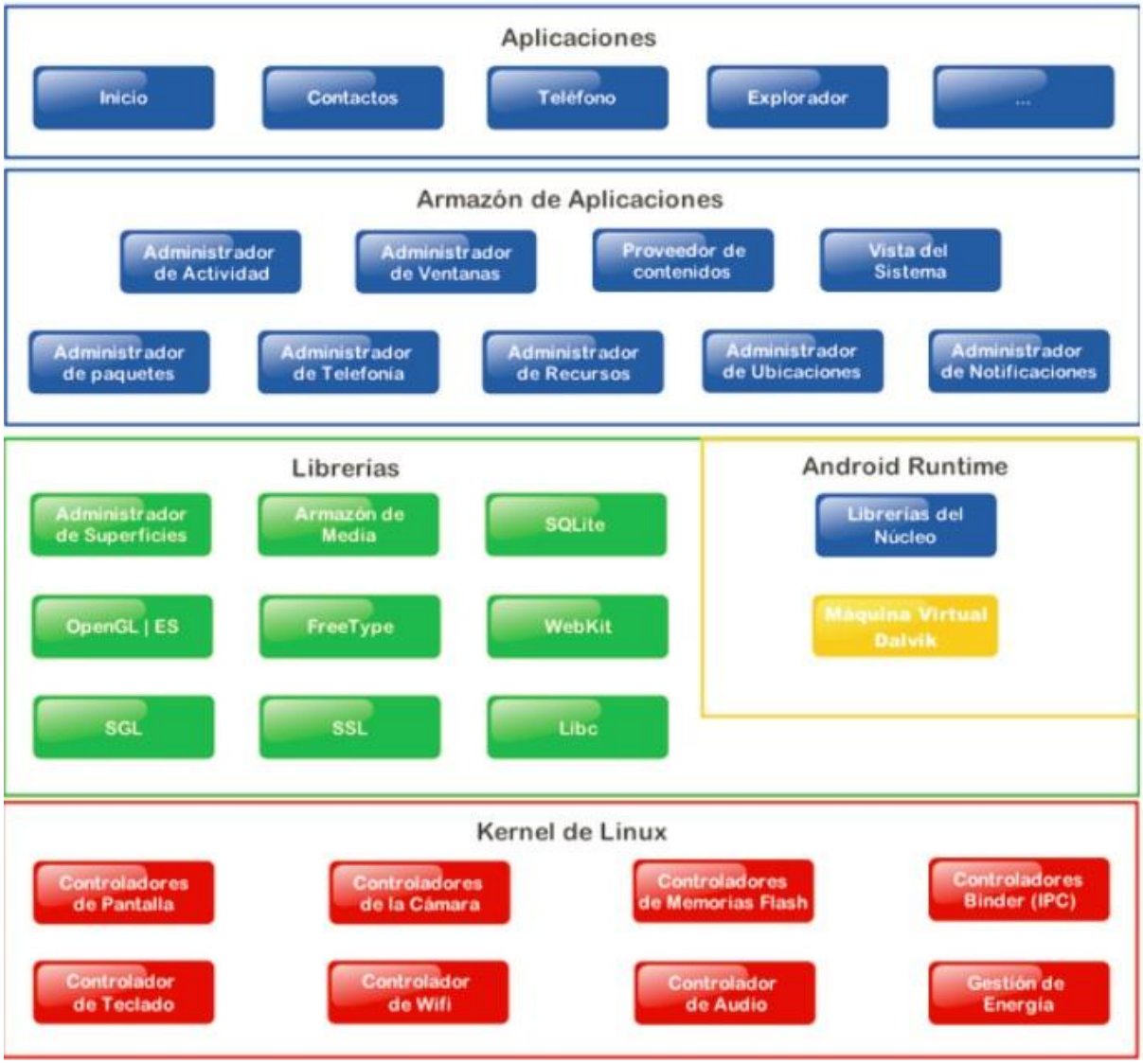

***Ilustración**4. Arquitectura de Android.(12)*

1.3.2 Android Studio. (12)

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android como:

- Un sistema de compilación basado en Gradle flexible.
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- Instant Run para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK.

- Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código.
- Gran cantidad de herramientas y frameworks de prueba.

Versión utilizada: 2.3.3

1.2.3 JAVA.

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta super computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes, que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos diez millones de usuarios reportados.(13)

1.3.4 Gradle.

Gradle es una herramienta de automatización de compilación de código abierto centrada en la flexibilidad y el rendimiento. Los scripts de compilación de Gradle se escriben usando Groovy o Kotlin DSL. Lea sobre las características de Gradle para saber qué es posible con Gradle.

- **Altamente personalizable:**Gradle se modela de una manera que se puede personalizar y ampliar de las formas más fundamentales.
- **Rápido:** Completa las tareas rápidamente reutilizando las salidas de las ejecuciones anteriores, procesando solo las entradas que cambiaron y ejecutando las tareas en paralelo.
- **Potente:** Es la herramienta de compilación oficial para Android y viene con soporte para muchos lenguajes y tecnologías populares.(14)

Versión utilizada: 3.4

1.3.5 XML.

XML (Extensible MarkupLanguage o Lenguaje de Marcado Extensible), se utiliza para representar información estructurada en la web, de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por muy diversos tipos de aplicaciones y dispositivos. Se clasifica como un lenguaje extensible porque permite a sus

usuarios definir sus propios elementos. Su objetivo principal es ayudar a los sistemas de información a compartir datos estructurados, particularmente a través de Internet, y se usa tanto para codificar documentos como para serializar datos.(15)

1.3.6 pgAdmin3.

Entorno de escritorio visual libre y de código abierto. Instalable en plataformas Linux, FreeBSD, Solaris, Mac OSX y Windows. Permite conectarse a bases de datos PostgreSQL que estén ejecutándose en cualquier plataforma. Está disponible en diferentes idiomas.

Facilita la gestión y administración de bases de datos ya sea mediante instrucciones SQL o con ayuda de un entorno gráfico. Permite acceder a todas las funcionalidades de la base de datos; consulta, manipulación y gestión de datos, incluso opciones avanzadas como manipulación del motor de replicación Slony-I.(16)

Versiónutilizada: 1.22.2

1.3.7 Pencil.

Pencil Project es una herramienta útil de creación de prototipos de GUI que equipa a las personas creativas para diseñar, dibujar, analizar y finalizar sus ideas utilizando una amplia gama de elementos, incluidas formas comunes, elementos web básicos, Sketchy GUI, plantillas y más. Estos se pueden exportar en formato PNG, SVG, HTML, PDF y ODT para aplicaciones en varios dominios de desarrollo con un plan artístico, pero técnicamente sólido, listo para terminar.(17)

Versión utilizada:2.0.3

1.3.8 Postman.

Es una herramienta que se utiliza, sobre todo, para el testing de API REST, aunque también admite otras funcionalidades que se salen de lo que engloba el testing de este tipo de sistemas.

Gracias a esta herramienta, además de testear, consumir y depurar API REST, podremos monitorizarlas, escribir pruebas automatizadas para ellas, documentarlasy simularlas.

Quizás sea una de las herramientas más utilizadas para hacer testing exploratorio de este tipo de sistemas. Puede que no sea la mejor forma de escribir pruebas automatizada, pero sin duda es una de las más favorables para equipos con poca experiencia en programación, y sobre todo para hacer testing de todo tipo en general de API REST.

Es importante destacar también que, aunque no sea una de las herramientas más famosas para documentar API REST, genera una documentación bastante interesante y bastante atractiva, con ejemplos y snippets de código, de forma que hace que sea muy fácil de entender cómo funciona una API determinada.(18)

Versión utilizada: 7.31.1

1.3.9Node.js

Es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. [8] Es el entorno de servidor web de código abierto más popular de la actualidad desarrollado por Ryan Dahl y JoyentInc (actualmente propiedad de Samsung Electronics). Al ser un entorno de tiempo de ejecución, Node.js ayuda a ejecutar códigos JavaScript fuera de un navegador. Utiliza el motor de Google Chrome para ejecutar JavaScript. En términos simples, ayuda a traducir los códigos de JavaScriptal lenguaje de nivel de máquina, por lo tanto, una máquina puede entender los códigos de JavaScript sin la necesidad de un navegador. Node.js en su conjunto facilita mucho el desarrollo de aplicaciones multiplataforma, del lado del servidor y de red. (19)

1.3.10 JSON.

JSON acrónimo de JavaScript ObjectNotation (Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, aunque hoy se considera un formato de lenguaje independiente. JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia. (20) Por lo antes expuesto, se emplea el formato JSON para el desarrollo del módulo en el paso de información entre la APK y el servicio web.

1.3.11 JWT.

JSON Web Token (JWT) es un estándar abierto (RFC 7519) que define una forma compacta y autónoma de transmitir información de forma segura entre las partes como un objeto JSON. Esta información se puede verificar y confiar porque está firmada digitalmente. Los JWT se pueden firmar usando un secreto (con el algoritmo HMAC) o un par de claves pública / privada usando RSA o ECDSA.

Aunque los JWT se pueden cifrar para proporcionar también secreto entre las partes, nos centraremos en los tokens firmados. Los tokens firmados pueden verificar la integridad de los reclamos que contiene, mientras que los tokens cifrados ocultan esos reclamos a otras partes. Cuando los tokens se firman utilizando pares de claves públicas / privadas, la firma también certifica que solo la parte que posee la clave privada es la que la firmó. (21)

1.3 Metodología de desarrollo.

Las metodologías ágiles poseen ciertas propiedades que las hacen totalmente aplicables al dominio de software móvil. Las metodologías que se pueden destacar son Adaptive Software Development, la familia de metodologías Crystal, el Método de Desarrollo de Sistemas Dinámicos (Dynamic SystemDevelopmentMethodology, DSDM), eXtremeProgramming(XP), Feature-DrivenDevelopment (FDD), Lean SofwareDevelopment, Scrum, Agile Modeling o PragmaticProgramming, Agile Model-DrivenDevelopment, Agile UnifiedProcess, RationalUnifiedProcess(22)

A continuación, se refieren algunas de las propiedades que presentan estas metodologías que son aplicables al desarrollo móvil.

Tabla 2. Comparativa entre las características básicas o bases (home ground) ágiles y los rasgos.(22)

Características ágiles	**Motivación lógica**	**Aplicable a tecnologías móviles.**
Alta volatilidad del entorno.	Debido a la frecuencia en el cambio que sufren los requisitos, se tienen menos necesidad de diseño y planificación inicial y mayor necesidad de desarrollos incrementables e iterativos.	Alta incertidumbre, entornos dinámicos, cientos de nuevos terminales cada año.
Equipos de desarrollos pequeños.	Capacidad de reacción más rápida, trabajo basado en la compartición de la información menos documentos.	La mayor parte de los proyectos de desarrollo móvil son de equipos pequeños.
Cliente identificable.	Conocimiento de los intereses de los clientes.	Potencialmente, hay un número ilimitado de usuarios finales, pero los clientes son fáciles de identificar.
Entornos de desarrollo orientados a objetos.	Mayoría de las herramientas de desarrollo ágil existen bajo plataformas orientadas a objetos.	Ejemplo, Java y C++ se usan, en algunos problemas en herramientas como refactorizaciones o primeros test.

Sistemas pequeños.	Menos necesidad de diseño inicial.	Las aplicaciones, aunque variables en tamaño, no suelen superar las 10.000 líneas de código.
Ciclos de desarrollo cortos.	Propósito de realimentación rápida.	Periodos de desarrollo de 1 a 6 meses.

Se ha seleccionado utilizar metodologías ágiles en el desarrollo de la aplicación Android para realizar las operaciones sobre los elementos de la planificación, por las características que presentan este tipo de desarrollo de software antes expuestas. Se presentará como propuestas en el desarrollo de la tesis, la selección de la metodología Variación AUP para la UCI, por la solicitud realizada por el centro FORTRES al cual tributa la tesis, en correspondencia con los lineamientos de desarrollo de software que posee la universidad.

1.3.1 Variación de AUP para la UCI.

La metodología de desarrollo a emplearse en los proyectos productivos de la Universidad de las Ciencias Informáticas (UCI) es Variación AUP-UCI. Dicha definición se basa en una variación de la metodología “Proceso Unificado Ágil (AUP por sus siglas en inglés) en unión con el modelo CMMI-DEV v1.3 que significa “CapacibilityMaturityModelIntegration”

La metodología Variación AUP-UCI consta de tres fases: Inicio donde se realizan las actividades relacionadas con la planeación del proyecto, permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto, la fase de Ejecución donde se ejecutan las actividades para desarrollar el software, se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y libera el producto. Por último, la fase de Cierre donde se analizan los resultados del proyecto y se realizan las actividades formales de cierre del proyecto.

Además, el ciclo de vida de los proyectos consta de siete disciplinas, Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación y Pruebas de aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto). Para el caso de este proyecto se estará utilizando el escenario número cuatro, en el cual se plantea: ¨Proyectos que no modelen negocio, solo pueden modelar el sistema con Historias de Usuario¨.(23)

1.4 Conclusiones del Capítulo 1.

Los conceptos definidos en un inicio servirán como base para el resto de la investigación. El estudio de los sistemas homólogos,aportó características y buenas prácticas para el desarrollo de interfaces para dispositivos móviles a tener en cuenta a lo largo del proceso de desarrollo. Las herramientas y tecnologías definidas actualmente son de las más utilizadas para el desarrollo de aplicaciones móviles,con lo que se garantiza que la aplicación tendrá soporte por un amplio período de tiempo. La metodología de desarrollo AUP en su variante para la UCI servirá de guía para el proceso de implementación del módulo. Quedando de esta forma conformado el marco teórico conceptual de la investigación.

CAPÍTULO 2

Capítulo 2: Análisis y diseño de la aplicación.

En el presente capítulo se lleva a cabo la descripción de la propuesta de solución definida para dar respuesta a la situación problémica existente. Tomando como referencia las pautas de requisitos, análisis y diseño e implementación que define la metodología AUP-UCI. También se analizan los patrones de diseño asociados a aplicaciones Android y la arquitectura del sistema.

2.1 Propuesta de solución.

Para dar solución a la problemática antes planteada, como parte delaaplicaciónmóvil a desarrollar, es necesario la implementación del módulo control docente, teniendo como principales funcionalidades: Registro de asistencia, Registro de evaluaciones, Gestionar control a clase y Gestionar grupos docentes. Como parte de las acciones para el desarrollo, al no contar con acceso a la plataforma de servicios de la universidad, se decide usar tecnologías alternativas, con el objetivo de simular la capa de servicios con la que se comunicará el sistema a implementar. Como una de las opciones seleccionada fue el uso del módulo de desarrollo de nodeJs denominado json-server este haciendo función como api rest brindará información lo más cercana posible a la realidad. La aplicación hereda los usuarios y permisos que ya existen hoy en AKADEMOS, a continuación, se hará mención de los implicados en la solución. El profesor, tendrá acceso a los registros de asistencia y evaluación de los grupos asignados, la secretaria docente es la que podrá realizar la gestión de los grupos docentes y el jefe de departamento podrá registrar los controles a clases.

No se pudo hacer uso de salvas pues la Dirección no cuenta con las bases de datos implicadas en la solución con información de prueba. No se procedió con la instalación de la plataforma de servicios (WSO2) vigente en la universidad por la complejidad de configuración y no poseer los requisitos mínimos de hardware que esta demanda para su optima ejecución.

2.2Especificación de Requisitos.

Los requisitos en el desarrollo de aplicaciones informáticas son uno de los pilares fundamentales y a la vez una de las principales causas que llevan los proyectos a la ruina. Están divididos en dos grupos principales, los funcionales, que definen las principales tareas que debe ser capaz de llevar a cabo el software y los no funcionales son pautas que deben seguir las funcionalidades y servicios brindados por la aplicación.(24)

2.2.1Requisitos Funcionales de la propuesta de solución.

Son declaraciones de los servicios y funciones que provee un sistema. Establecen a grandes rasgos o de manera descriptiva qué es lo que debe hacer el sistema. En ocasiones también explican qué no debe hacer el sistema ante determinadas situaciones específicas. (24)

A continuación, se muestra una tabla con los requisitos funcionales obtenidos luego de un proceso de levantamiento de requisitos mediante una serie de entrevistas con el cliente. Para cada uno de los requisitos se definió una complejidad y una prioridad en cuanto a su desarrollo.

Tabla 3. Requisitos Funcionales.

No	Nombre	Descripción	Complejidad	Prioridad
RF1	Gestionar Registro de Asistencia.	El usuario registra el estado presente o ausente de los estudiantes del grupo que atiende.	Alta	Alta
RF2	Visualizar Registro de Asistencia.	Permite al usuario visualizar una lista de los registros de asistencia previamente almacenados en el sistema.	Baja	Baja
RF3	Eliminar Registro de Asistencia.	Permite al usuario eliminar un registro de asistencia previamente almacenado en el sistema.	Media	Media
RF4	Gestionar Registro de Evaluaciones.	El usuariodefine una nota para cada uno de los estudiantes del grupo docente asignado.	Alta	Alta
RF5	Visualizar Registro de Evaluaciones.	Permite al usuario visualizar una lista de los registros de evaluaciones previamente almacenados en el sistema.	Baja	Baja
RF6	Eliminar Registro de Evaluaciones.	Permite al usuario eliminar un registro de evaluaciones previamente almacenado en el sistema.	Media	Media
RF7	Listar Registros	Se listan el registro de todos los estudiantes que pertenecen al profesor autenticado previamente en el sistema.	Baja	Baja
RF8	Listar Control a Clases.	Se listan los controles a clase realizados a cada uno de los profesores por el usuario	Baja	Baja

		autenticado según su nivel de acceso al sistema.		
RF9	Detalles de Control a Clases.	Se muestran los detalles de un control a clases previamente seleccionado.	Baja	Baja
RF10	Modificar Control a Clases.	El usuario puede modificar la información de un control a clases previamente seleccionado.	Media	Media
RF11	Crear Grupo Docentes.	El usuario es capaz de crear un nuevo grupo docente en el sistema.	Media	Media
RF12	Modificar Grupo Docente.	El usuario modifica un grupo docente existente en el sistema.	Media	Media
RF13	Eliminar Grupo Docente.	El usuario elimina un grupo docente existente en el sistema.	Media	Media
RF14	Detalles de Grupo Docente.	Se muestran los detalles de un grupo docente previamente seleccionado.	Baja	Baja
RF15	Mostrar Registro Docente General.	Muestra el registro de asistencia y el registro de evaluaciones de cada estudiante que atiende el profesor autenticado en el sistema.	Baja	Baja

2.2.2Requisitos No Funcionales de la propuesta de solución.

Son propiedades o cualidades que el producto debe tener. No se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes del mismo que hacen al producto atractivo, usable, rápido, confiable, etc. Normalmente son rasgos que se aplican al sistema en su totalidad y no a funciones específicas. (25)

A continuación, se enumeran los requisitos no funcionales obtenidos luego de un proceso de levantamiento de requisitos mediante una serie de entrevistas con el cliente:

- **Interfaz: RnF1** El módulo deberá ser desarrollado empleando las pautas que definencorrecto desarrollo de interfaces en Android(26):
  - ✓ El Área de acción debe ser como mínimo de 44dp[2].
  - ✓ El tamaño de fuente debe ser como mínimo de 12dp.

[2]La definición técnica de ¨dp¨ es Density-Independient Pixel (Píxel de Densidad Independiente), hace referencia a la cantidad de pixeles que son necesarios para ocupar un área en la pantalla independientemente de la densidad que esta tenga.(40)

- ✓ No debe existir predominio de más de tres colores principales en la aplicación, estos colores deben representar con claridad el negocio al que responde la aplicación (naranja, amarillo y blanco en este caso, pues son los colores que definen AKADEMOS).

- **Confiabilidad/ Fiabilidad: RnF2** Garantizar la integridad y confidencialidad de la información mediante mecanismos de seguridad. Esto se garantiza con el mecanismo de autenticación utilizado (JWT) y los protocolos de comunicación con los servicios a utilizar.
- **Confiabilidad/ Tolerancia al error: RnF3** Mostrar la información necesaria del error.
- **Usabilidad/ Instructibilidad: RnF4**La interfaz debe ser intuitiva, y lo más parecida posible a AKADEMOS, ya que existe un grupo de usuarios que son profesores de avanzada edad, y estadísticamente este grupo de usuarios tiene un alto rechazo al cambio y una alta dificultad a la hora de interactuar con nuevas tecnologías.
- **Usabilidad/ Instructibilidad: RnF5** Mostrar elementos de apoyo a la navegación.
- **Usabilidad/ Operatividad: RnF6** Mostrar funcionalidades de acuerdo a los privilegios.
- **Disponibilidad: RnF7** Disponibilidad del módulo las 24 horas del día.
- **Software: RnF8** El cliente debe tener sistema operativo Android 5.0 o superior.
- **Soporte: RnF9** El módulo contará con toda la documentación definida en la presente investigación durante el proceso de desarrollo.

2.3 Historias de Usuario.

Las Historias de Usuario (HU) es un artefacto generado por metodologías ágiles como Scrum, XP y la variación de AUP-UCI específicamente en su escenario 4, en el cual se basa este proyecto. Son utilizadas con el fin de especificar los requisitos del software. A través de un conjunto de tablas se describen brevemente las características que desea el cliente. Contienen la información suficiente para que los desarrolladores puedan producir una estimación razonable del esfuerzo necesario para su implementación. Su contenido está estructurado de forma tal que se puedan descomponer en tareas para su posterior desarrollo entre 2 y 4 semanas. (27)

Con el objetivo de presentar los requisitos funcionales concisa y detalladamente, se realizaron las historias de usuario (HU) lo que permite hacer una descripción de los requisitos funcionales de la aplicación. A continuación, se muestra la HU del Requisito funcional Listar registros. Una vez que el profesor se autentica en la aplicación, y accedea la

opción "Registro Docente" del menú principal, se le muestra una vista con la lista de los registros de los grupos que este tiene asociado como profesor de la asignatura que imparte.

Tabla 4. Historia de Usuario Listar Registros.

Historia de Usuario	
Código:HU7	**Nombre Historia de Usuario:** Listar Registros
Modificación de Historia de Usuario Número: *ninguna*	
Referencia: RF7	
Programador: Mauro Sergio Sotolongo Montes	**Iteración Asignada:** *Primera*
Prioridad: Alta	**Tiempo Estimado:***1 semanas*
Riesgo en Desarrollo: *Medio*	
Descripción: La historia de usuario permite visualizar en forma de tabla los registros que se encuentran en el sistema. Existen tres campos que permiten filtrar los registros que se mostrarán en la tabla. • Facultad: Filtra la búsqueda para que solo se muestren los registro que pertenecen a la facultad seleccionada. El listado de facultades se obtiene de las facultades que se encuentran activas en el sistema. • Grupo: Filtra la búsqueda para que solo se muestren los registro que pertenecen al grupo administrativo seleccionado. El listado de grupos administrativos se obtiene de los grupos administrativos que se encuentran activos en el sistema. • Asignatura: Filtra la búsqueda para que solo se muestren los registro que pertenecen a la asignatura seleccionada. El listado de asignaturas se obtiene de las asignaturas que se encuentran activas en el sistema. • El usuario puede pulsar sobre una fila para que se muestren las opciones a las que tiene acceso del registro perteneciente a la fila seleccionada.	
Rol: Profesor	
Prototipo de interfaz: **IU. Listar Registros**	

2.4Arquitectura de Desarrollo de Software Cliente – Servidor a tres capas.

Los sistemas cliente/servidor están construidos de tal modo que la base de datos puede residir en un equipo central, llamado servidor y ser compartida entre varios usuarios. Los usuarios tienen acceso al servidor a través de una aplicación cliente(22):

En un sistema cliente/servidor de varios componentes, la lógica de la aplicación de cliente se ejecuta en dos capas:

- El cliente reducido se ejecuta en el equipo local del usuario y se encarga de presentar los resultados al usuario.
- La lógica del negocio se encuentra en aplicaciones de servidor que se ejecutan en un servidor. Los clientes reducidos solicitan funciones a la aplicación de servidor, que, a su vez, es una aplicación multiproceso capaz de operar con varios usuarios simultáneos. La aplicación de servidor es la que abre las conexiones con el servidor de la base de datos y se puede ejecutar en el mismo servidor que la base de datos, o se puede conectar a través de la red con otro servidor que opere como servidor de basede datos.

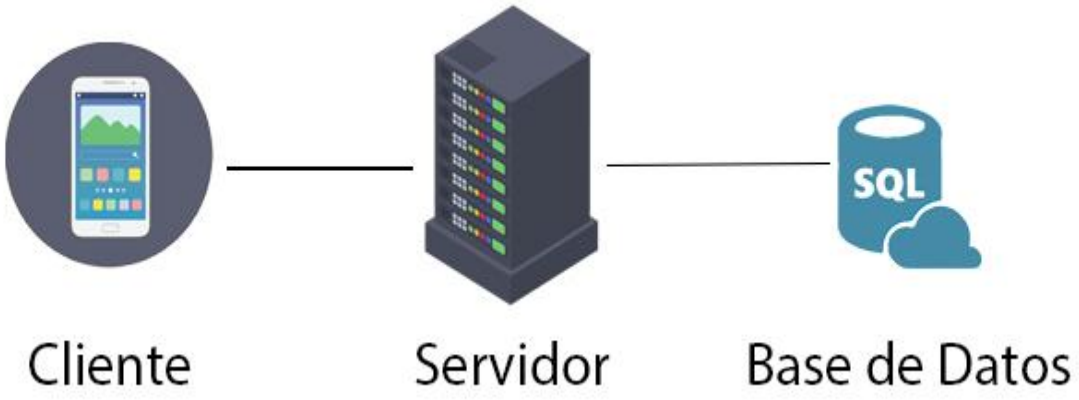

Ilustración 5. *Arquitectura Cliente-Servidor.*

Ventajas de esta arquitectura para el sistema que se desarrollará:

1. Al estar gestionada la lógica del negocio en el servidor, se consigue que:
   - La aplicación requiera una menor cantidad de espacio al ser instalada.
   - Consuma muy pocos recursos.
2. En cuestiones de seguridad de los datos, en la parte del cliente no existen sentencias SQL que permitan tener noción de que estructura tienen las tablas en la base de datos, y el mismo no tiene interacción directa con los datos, sino que es el proveedor de servicios quien realiza esta función en su logar.
3. Un cambio en la manera de representar los datos en la base de datos no tendría repercusión en la aplicación cliente, por lo que no sería necesario obligar a los usuarios a actualizar la APK para poder usar el sistema.

2.5 Patrones.

Un patrón es un tipo de tema de sucesos u objetos recurrentes. Puede definirse como aquella serie de variables constantes, identificables dentro de un conjunto mayor de datos los cuáles se repiten de una manera predecible. Una plantilla o modelo puede usarse para generar objetos o partes de ellos, especialmente si los objetos que se crean tienen lo suficiente en común para que se infiera la estructura del patrón fundamental, en cuyo caso, se dice que los objetos exhiben un único patrón. En la ingeniería del software, un patrón constituye el apoyo para la solución a los problemas más comunes que se presentan durante las diferentes etapas del ciclo de vida de un software.(28)

2.5.1 Patrón Arquitectónico.

Los patrones arquitectónicos se utilizan para expresar una estructura de organización base o esquema para un software. Proporcionando un conjunto de sub-sistemas predefinidos,

especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos. (29)(22)

2.6Patrón de Arquitectura del Sistema.

La arquitectura MVC propone, independientemente de las tecnologías o entornos en los que se base el sistema a desarrollar, la separación de los componentes de una aplicación en tres grupos (o capas) principales: el modelo, la vista, y el controlador, y describe cómo se relacionarán entre ellos para mantener una estructura organizada, limpia y con un acoplamiento mínimo entre las distintas capas.(30)

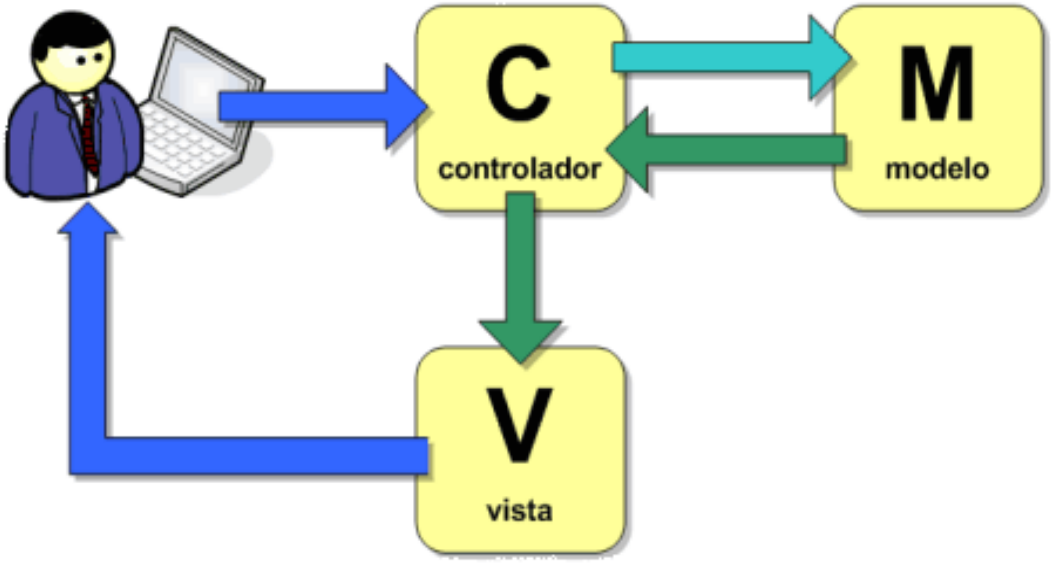

Ilustración 6. *Relación Modelo-Vista-Controlador.*

- **El Modelo**.

  En la capa Modelo encontraremos siempre una representación de los datos del dominio, es decir, aquellas entidades que nos servirán para almacenar información del sistema que estamos desarrollando. Ejemplos de modelos presentes en el sistemaserían las clases Estudiante, Profesor, Evaluación.

- **La Vista.**

  Los componentes de la Vista son los responsables de generar la interfaz de nuestra aplicación, es decir, de componer las pantallas, páginas, o cualquier tipo de resultado utilizable por el usuario o cliente del sistema. Cuando las vistas componen la interfaz de usuario de una aplicación, deberán contener los elementos de interacción que permitan al usuario enviar información e invocar acciones en el sistema, como botones, cuadros de edición o cualquier otro tipo de elemento, convenientemente adaptados a la tecnología del cliente. En el

sistema, son las *activities*en su componente .XML las encargadas de esta función.

- **El Controlador.**

  La misión principal de los componentes incluidos en el Controlador es actuar como intermediarios entre el usuario y el sistema. Serán capaces de capturar las acciones de este sobre la Vista, como puede ser la pulsación de un botón o la selección de una opción de menú, interpretarlas y actuar en función de ellas. Realizarán también tareas de transformación de datos para hacer que los componentes de la Vista y el Modelo se entiendan. Así, traducirán la información enviada desde la interfaz a objetos que puedan ser comprendidos por el Modelo.

  De igual manera que en la Vista, son las *activities* las encargadas de llevar a cabo dichas tareas, solo que en esta ocasión a través del componente .JAVA.

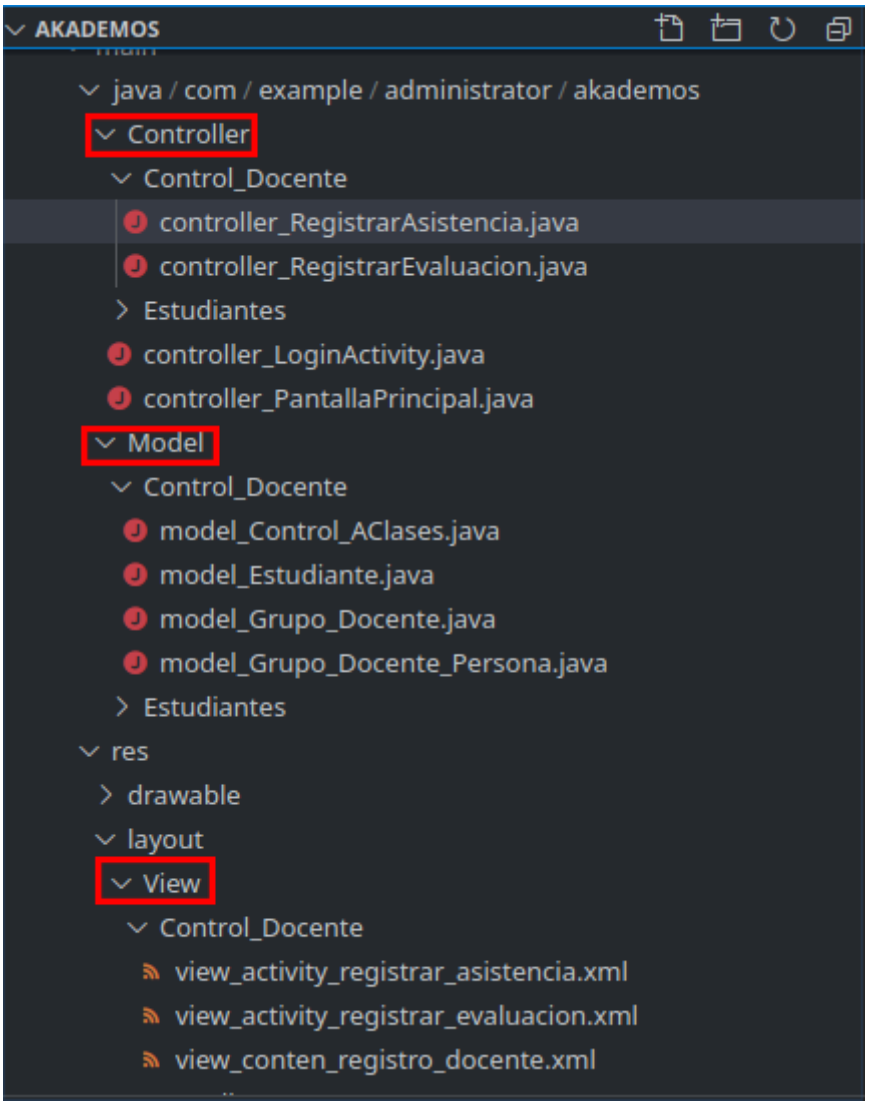

Ilustración 7. *Estructura del módulo Control Docente siguiendo el patrón arquitectónico MVC.*

2.7Patrones de Diseño del Sistema.

Un patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces. Los patrones de diseño se pueden utilizar en cualquier lenguaje de programación orientado a objetos. El patrón de diseño proporciona una solución probada a un problema común, documentada en un formato coherente.(31)

2.7.1Patrones de asignación de responsabilidad (GRASP).

Los patrones GRASP constituyen un apoyo para la enseñanza que ayuda a uno a entender el diseño de objetos esencial, y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la comprensión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades.(32)(33)

Los patrones GRASP pueden ser:

- Experto: es el principio básico de asignación de responsabilidades. Determina cuál es la clase que debe asumir una responsabilidad a partir de la información que posee. O sea, la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).Este patrón es evidenciado en la definición de las clases de acuerdo a las funcionalidades que deben realizar a partir de la información manejada dentro del componente, como por ejemplo las clases controladoras.
- Creador:Ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. Se identifica al asignar a la clase B la responsabilidad de crear una instancia de clase A si se cumple uno o más de los casos siguientes:
    - B agrega objetos de A.
    - B contiene objetos de A.
    - B registra instancias de objetos de A.
    - B utiliza más estrechamente objetos de A.
    - B tiene los datos de inicialización que se pasarán a un objeto de A cuando sea creado (por tanto, B es un Experto con respecto a la creación de A).
    - B es un creador de los objetos A. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador.

Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien, el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización. Se utilizó este patrón en cada clase controlador.

- Bajo Acoplamiento:Medida de la fuerza con que una clase está conectada a otras clases. Propone tener las clases lo menos ligadas entre sí, de forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases. Potencia la reutilización de código y disminuye la dependencia. Se utilizó este patrón en la clase MainActivity.java. Esta clase al ser la encargada de mostrar en pantalla la vista principal de la aplicación, solamente depende de las clases indispensables del framework de Android para realizar su tarea.

2.7.2Patrones GOF(Gang of Four).

Lo patrones GOF son útiles durante el diseño de objetos. Los patrones GoF pueden ser de tres tipos: de creación, comportamiento y estructurales.

Los patrones de comportamiento plantean la interacción y cooperación entre las clases. Estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal.(34)

En la solución propuesta se utilizó el patrón de comportamiento Strategy (Estrategia) que permite disponer de varios métodos para resolver un problema y elegir cuál de ellos utilizar en tiempo de ejecución. En la implementación realizada se utiliza la clase GenerarPDF.java. Esta clase recibe la petición de generar un archivo PDF desde diferentes funcionalidades de la aplicación como pueden ser generar un documento PDF del Registro de Evaluaciones, del Registro de Asistencias o de un Control a Clases. Indistintamente desde que funcionalidad se realice la petición de generar un archivo PDF, la clase antes mencionada es capaz de identificar qué tipo de contenido se desea generar y en dependencia del tipo, utiliza un modelo y un método que permiten generar este archivo, sin necesitar tener una clase generadora de PDF para cada uno de los tipos de contenidos.

Los patrones estructurales son aquellos que se encargan del diseño del software,y solucionan problemas de composición (agregación) de clases y objetos.(35)

En la solución propuesta se utilizó el patrón estructural Adapter (Adaptador) que permite que un objeto sea utilizado por otra clase que de otra manera no pudiera ser utilizado. En la

implementación para representar las listas en las interfaces se hace uso de la clase RecyclerView que es propia del framework de Android. Para poder mostrar la información de los registros docentes en el RecyclerView fue necesario implementar la clase AdapterRegistroDocente.java que hereda de la clase Adapter.java, la cual es la encargada de convertir los datos obtenidos luego de realizar la consulta en elementos de interfaz de usuario.

2.8 Modelo de datos.

El diseño de una base de datos consiste en definir la estructura de los datos que debe tener un sistema de información determinado. Para ello se suelen seguir por regla general unas fases en el proceso de diseño, definiendo para ello el modelo conceptual, el lógico y el físico.En el modelo relacional las dos capas de diseño conceptual y lógico, se parecen mucho. Generalmente se implementan mediante diagramas de Entidad/Relación (modelo conceptual) y tablas y relaciones entre éstas (modelo lógico). Este es el modelo utilizado por los sistemas gestores de datos más habituales (SQL Server, Oracle, MySQL). (36)

A continuación, se muestra el modelo de datos que evidencia las tablas a las que los servicios web brindarán acceso a la aplicación Android. Es importante destacar que este modelo de datos pertenece al actual sistema AKADEMOS y que la aplicación Android deberá seguir este modelo.La tabla tb_dgrupo_docente contiene la información referente al grupo, asignatura, año y semestre, esta es la entidad principal que regirá el resto de los registros. En la tabla tb_dcontro_clases es donde se registra la información obtenida en un control a clase. En la tablatb_rgrupo_docente_persona es donde se hacen persistentes los estudiantes y profesores que pertenecen a un grupo docente. La tabla tb_dprofesor_principal almacena información del módulo para el profesor y la tabla tb_estudiante contiene la información básica del estudiante.

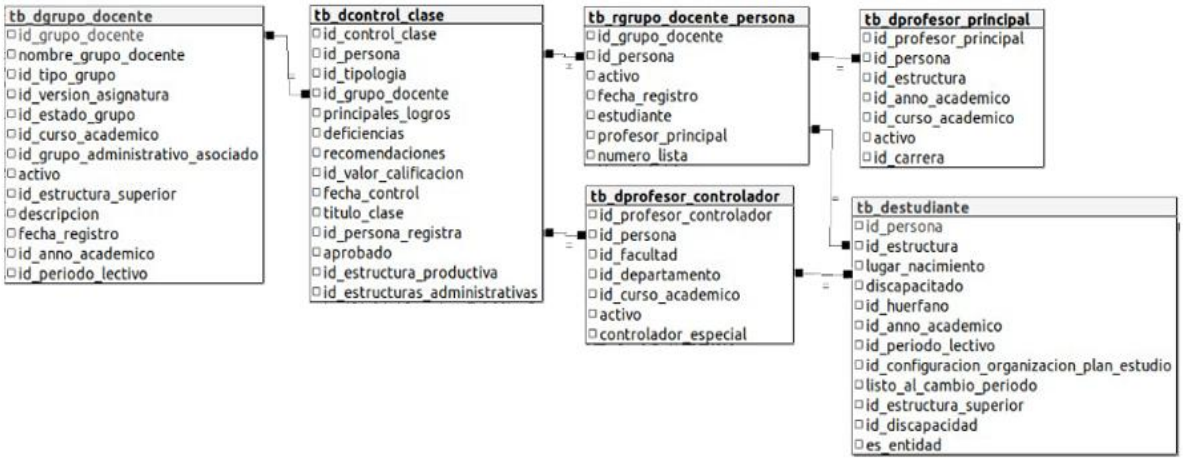

Ilustración 8. *Modelo de datos del módulo Control Docente.*

2.9 Seguridad del Sistema.

La aplicación del módulo control docente forma parte de un proyecto que tiene el objetivo de integrar varios módulos del sistema de gestión académica AKADEMOS. Los resultados de este trabajo de diploma y los del trabajo de diploma que responde al módulo estudiantes tributan en conjunto a este proyecto.

La aplicación del módulo control docente hace uso de la seguridad que es común para todo el proyecto. Existe una vista de autenticar que es quien accede al servicio de seguridad de la UCI. Este hace uso del servicio que genera un JWT para obtener el token de seguridad de autenticación del usuario, este contiene la información del nivel de acceso que posee el usuario autenticado en el sistema. La aplicación muestra una vista denominada Pantalla Principal, en la cual se presentan todos los módulos que incluye el sistema, solo aparecerán activos los que se encuentra permitidos para el nivel de acceso del usuario en curso. Para poder analizar la información devuelta en el token es necesario decodificar el archivo JSON obtenido, para ello es necesario hacer uso de la librería para java JJWT. La información obtenida en este proceso de decodificación es enviada a través de las *activities* como parámetros del método *Intent*, para que puedan ser usadas por los diferentes módulos cuando lo necesiten.

2.10Conclusiones del Capítulo 2.

En el presente capítulo se realizó el análisis y diseño de la aplicación.La representación de las historias de usuario posibilitó idealizar desde la perspectiva del usuario final, los valores que aportan los componentes del sistema, lo que permitirá desarrollar con mayor facilidad la solución propuesta pensando en el usuario final.El levantamiento de requisitos propició identificar, describir y clasificar los requerimientos funcionales y no funcionales que debe satisfacer la solución.Se identificaron 15 requisitos funcionales y 9 no funcionales.La utilización de patrones de diseño y arquitectura tales como el patrón Cliente-Servidor, MVC, los patrones GRASP y GOF permitieron diseñar una solución ágil, flexible y escalable.A partir del análisis, el diseño realizado y los artefactos generados quedan sentadas las bases para la implementación, pruebas y validación de la solución propuesta.

Capítulo 3: Implementación y pruebas.

En el presente capítulo se lleva a cabo la representación del proceso de despliegue para la aplicación Android, mostrando la comunicación que deberá existir entre la aplicación en los dispositivos y los servidores. Se evidenciará el resultado obtenido luego de concluido el proceso de implementación de la aplicación, que fue realizado respetando la prioridad definida para cada requisito funcional, presentándose algunos de los principales prototipos de interfaces, que fueron diseñadas siguiendo las pautas que proponenlos requisitos no funcionales referentes a interfaz y usabilidad. Se realizará la documentación del proceso de pruebas llevado a cabo, garantizando la calidad del producto final obtenido.

3.1 Diagrama de Despliegue.

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos.

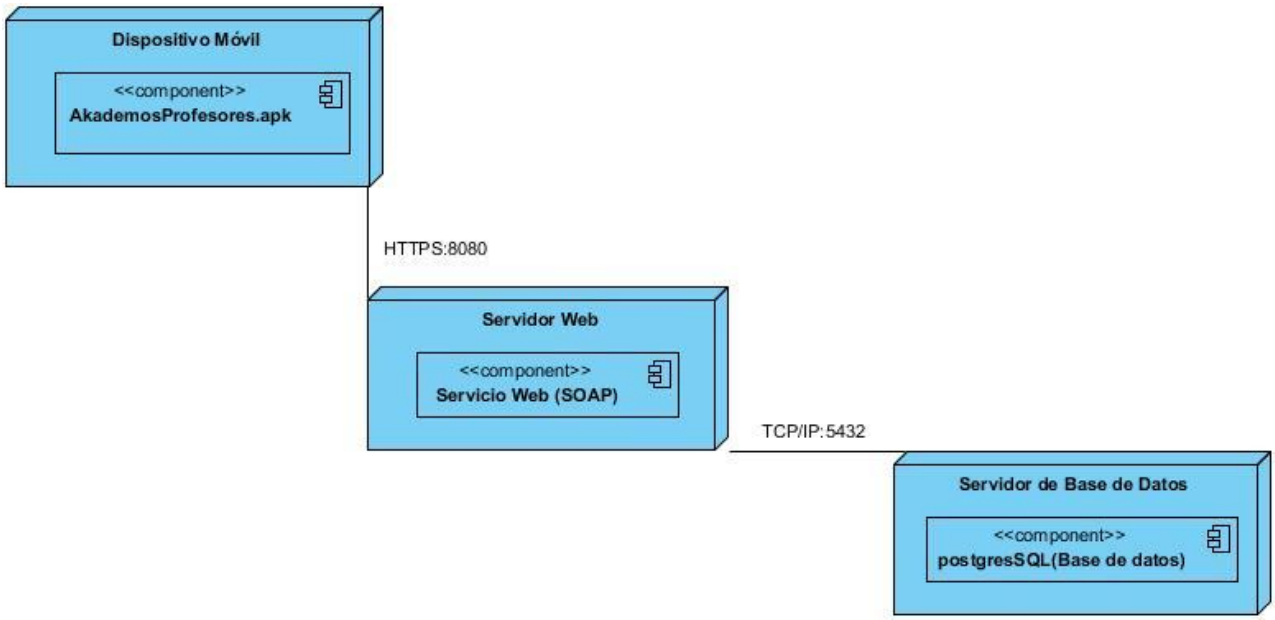

Ilustración 9. *Diagrama de Despliegue.*

Nodo Dispositivo Móvil: En este nodo se ejecuta la aplicación Android, la cual hace uso de los servicios webs a través del protocolo HTTPS.

Nodo Servidor Web: Es el host de los servicios web en cargados de dar solución a la lógica del negocio, sirviendo de intermediario entre los otros dos nodos.

Nodo Servido de Base de Datos: En este nodo se encuentran almacenados los datos que conforma la información del sistema, usuarios, registros, asignaturas, grupos administrativos, por solo citar algunos ejemplos.

3.2Implementación de la solución.

Una vez concluida la implementación parcial bajo las condiciones anteriormente descritas, se puede verificar que la aplicación está lista para realizar la integración con la plataforma de servicios de la universidad.

La interfaz (Ilustración 7) Muestra la vista del Menú Lateral de la Pantalla Principal de la aplicación.

Ilustración 10 *Menú Lateral.*

La interfaz (Ilustración 8) Muestra la vista de la Pantalla Principal de la aplicación, donde aparecerán todas las funcionalidades a las que el usuario autenticado tenga acceso.

Ilustración 11 *Vista de la Pantalla Principal.*

La interfaz (Ilustración 9) Muestra la vista de la funcionalidad Registrar asistencia de la aplicación.

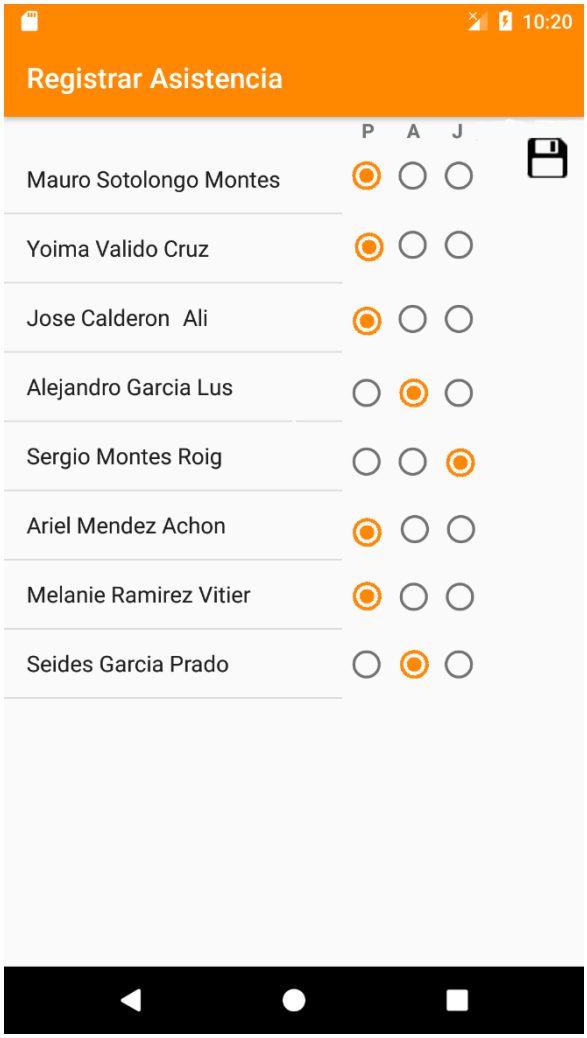

Ilustración 12 *Vista del RF Registrar Asistencia.*

3.3Pruebas de Software.

Las pruebas de software es seguramente la actividad más común de control de calidad realizada en los proyectos de desarrollo o mantenimiento de aplicaciones y sistemas. Las pruebas de software se definen como una actividad en el cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificada, los resultados se observan y registran y se realiza una evaluación de algún aspecto. Las pruebas pueden clasificarse en funcionales y no funcionales:(37)

Tipos de **Pruebas Funcionales**:Podemos considerar el proceso de pruebas funcionales como un proceso donde se va probando inicialmente lo de más bajo nivel y se van integrando y probando paulatinamente componentes hasta lograr un sistema completo totalmente probado.(38)

- Pruebas unitarias.
- Pruebas de componentes.
- Pruebas de integración.
- Pruebas de sistema.
- Pruebas de humo.
- Pruebas de aceptación.
- Pruebas de regresión.

Tipos de **Pruebas No Funcionales**: Una prueba no funcional es una prueba cuyo objetivo es la verificación de un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema (requisitos no funcionales) como por ejemplo la disponibilidad, accesibilidad, usabilidad, mantenibilidad, seguridad, rendimiento.(38)

- Pruebas de compatibilidad.
- Pruebas de seguridad.
- Pruebas de estrés.
- Pruebas de usabilidad.
- Pruebas de rendimiento.
- Pruebas de internacionalización y localización
- Pruebas de escalabilidad.
- Pruebas de mantenibilidad.
- Pruebas de instalabilidad.
- Pruebas de portabilidad

Con el objetivo de llevar a cabo la evaluación de la propuesta de solución actual, teniendo en cuenta las características y los propósitos del sistema, se seleccionó la aplicación de dos tipos de pruebas correspondientes a las pruebas de aceptación, pruebas unitarias (Caja Blanca) y pruebas de integración.

3.3.1Pruebas de Aceptación.

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

Las pruebas de aceptación son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es

recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información. (39)

Se muestra un ejemplo de prueba de aceptación realizada a la Historia de Usuario No.3 Listar Registros en el último ciclo de iteración, una vez instalada la aplicación en la terminal del profesor que brindó voluntario para realizar la prueba, se procede a autenticarse en el sistema, después de acceder a la funcionalidad se procede a realizar las acciones que se reflejan en la tabla 5 obteniendo resultados favorables.

Tabla 5 Ejemplo de Prueba de Aceptación.

Caso de Prueba de Aceptación	
Código: **HU15_P2**	**Historia de Usuario:** No.3 Listar Registros
Nombre: **Listar Registros**	
Descripción: **Prueba la funcionalidad de realizar filtrados en la lista de registro.**	
Condiciones de ejecución: **El usuario debe estar autenticado.**	
Entrada/Pasos de ejecución: **El usuario selecciona la opción *"Registro Docente"*.** **El usuario selecciona la opción *"Listar Registros Docentes".*** **En usuario selecciona en cada uno de los spinner que desea filtrar, ya sea facultad, grupo académico o asignatura.**	
Resultado esperado: **Muestra los estudiantes de la tabla según el filtrado que se realice.**	
Evaluación de la prueba: **Prueba satisfactoria.**	

A continuación, se muestra una tabla resumen que recoge la cantidad de no conformidades obtenidas en cada ciclo de iteración de las pruebas de aceptación realizadas a la aplicación. La cual evidencia que se logran mitigar todos los errores encontrados en cada iteración.

Tabla 6. Resultado de las pruebas realizadas por cada iteración.

Cantidad de No Conformidades/ Iteración	Iteración 1	Iteración 2	Iteración 3
No Conformidades	14	7	Ninguna

3.3.2 Pruebas Unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. (40)

Las pruebas de caja blanca, denominadas a veces pruebas de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba.Con la aplicación de este método se garantiza que se verifique por lo menos una vez todos los caminos independientes de cada módulo, que se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas, que se ejecuten todos los bucles en sus límites y con sus límites operacionales y se entrenen las estructuras internas de datos para asegurar su validez. (22)

Se hace uso de la técnica del camino básico que permite derivar casos de prueba a partir de un conjunto de caminos independientes por los cuales puede circular el flujo de control.

Pasos para el desarrollo de la técnica del Camino básico:

1. Se escoge el diseño o el código de la aplicación como base, y se dibuja el correspondiente grafo de flujo. Se obtiene la complejidad ciclomática del grafo de flujo.
2. Se determina la complejidad ciclomática del grafo resultante, posteriormente se determinan los casos de pruebas que permitan la ejecución de los caminos anteriores.
3. Se establece un conjunto básico de caminos linealmente independientes.
4. Se elaboran los casos de pruebas que permitirán la ejecución de cada camino del conjunto básico.

Notación del grafo de flujo:

- **Nodo del grafo de flujo(N):** Es un círculo que representa las sentencias procedimentales.
- **Aristas(A):** Son las flechas o enlaces que representan el flujo de control del grafo.
- **Regiones:** Son las áreas delimitadas por las aristas y nodos. Al realizar el conteo de las regiones también se incluye el área exterior del grafo.

- **Complejidad ciclomática:** Se calcula a partir del grafo de control del software bajo análisis y mide, a grandes rasgos, el número de caminos independientes que pueden ocurrir en la ejecución, entre el inicio y el fin del software.
  Se define como: V (G)= A-N+2.

A continuación, se muestra un fragmento de código perteneciente al algoritmo desarrollado:

```
public estudiante[]  BuscarPorNombre(estud, nombre){
    int j = 0;
    estudiante[] estudiantesF = [];

    for(int i = 0, i < estud.length, i++){
        if(estud[i].nombre() != null && estud[i].nombre() != ""){
            nombrecoleccion = SustituirAcentos(estud[i].nombre().toLoerCase());
            nombreenvido = SustituirAcentos(nombre.toLoerCase());
            if(nombrecoleccion.indexOf(nombreenvido) != -1 && nombrecoleccion.indexOf(nombreenvido) != ""){
                estudiantesF[j++] = estud[i];
            }
        }
    }
    return estudiantesF;
}
```

Ilustración 13. *Fragmento de código para buscar por nombre.*

Posteriormente se procede a la elaboración del grafo de flujo teniendo en cuenta dicha enumeración.

Complejidad ciclomática

V (G)= A-N+2

V (G)= 10 - 8 +2 = 4

Caminos independientes

1, 2, 3, 8

1, 2, 3, 4, 3, 8

1, 2, 3, 4, 5, 6, 3, 8

1, 2, 3, 4, 5, 6, 7, 3, 8

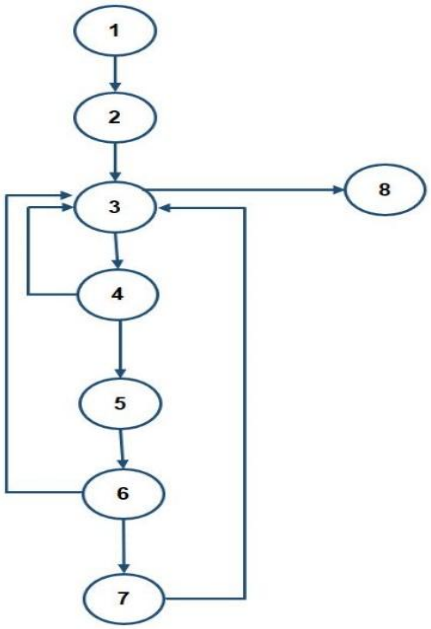

Ilustración 14. *Representación del grafo de flujo del camino básico*

Cada camino independiente es un caso de prueba a realizar, de forma que se garantiza que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. En el caso anterior se calcularon 4 caminos básicos, por tanto, surge la necesidad de hacer igual número de casos de prueba. A continuación, se muestra los 4 casos de pruebas realizados.

Tabla 7. Casos de prueba realizados para el camino base.

Caso de prueba 1 para el camino básico	
Descripción Mostrar la tabla sin datos si la lista de estudiantes es vacía.	
Condición de ejecución Base de datos vacía.	
Datos de entrada	Nombre y apellidos.
Tipo de datos esperado	
Evaluación del caso de Prueba: Satisfactoria	
Caso de prueba 2 para el camino básico	
Descripción Mostrar la tabla sin datos si el campo nombre de la entidad estudiantes es vacío o nulo.	
Condición de ejecución	

Atributo nombre es nulo o vacío.	
Datos de entrada	Nombre y apellidos.
Tipo de datos esperado	
Evaluación del caso de Prueba: Satisfactoria	
Caso de prueba 3 para el camino básico	
Descripción Mostrar la tabla sin datos si no hay coincidencia de nombres.	
Condición de ejecución Nombres inexistentes	
Datos de entrada	Nombre y apellidos.
Tipo de datos esperado	
Evaluación del caso de Prueba: Satisfactoria	
Caso de prueba 4 para el camino básico	
Descripción Mostrar la tabla con datos de los estudiantes	
Condición de ejecución Nombres coincidentes	
Datos de entrada	Nombre y apellidos.
Tipo de datos esperado	
Evaluación del caso de Prueba: Satisfactoria	

3.3.3 Pruebas de Integración.

Las pruebas de integración son una técnica sistemática para construir la arquitectura del software y, al mismo tiempo, realizar pruebas para descubrir errores asociados con la interconexión. El objetivo es tomar componentes probados por unidades y construir la estructura del programa que se diseñó.

A menudo hay una tendencia a intentar la integración no incremental; es decir, para construir el programa usando un enfoque de "Big Bang". Todos los componentes se combinan de antemano. Todo el programa se prueba como un todo. Y el caos suele resultar. Se encuentra un conjunto de errores. La corrección es difícil porque el aislamiento de las causas se complica por la vasta extensión de todo el programa. Una vez que se corrigen estos errores, aparecen otros nuevos y el proceso continúa en un bucle aparentemente interminable.

La integración incremental es la antítesis del enfoque del “Big Bang”. El programa se construye y se prueba en pequeños incrementos, donde los errores son más fáciles de

aislar y corregir. Es más probable que las interfaces se prueben completamente y se puede aplicar un enfoque de prueba sistemático. (22)

Se emplea el método de integración incremental por las facilidades que brinda al desarrollador de realizar pruebas al programa menos complejas y obteniendo mejores resultados. A continuación, se presentan dos estrategias del método seleccionado.

Estrategias de integración

Top-down (descendente): La prueba de integración descendente es un enfoque incremental para la construcción de la arquitectura del software. Los módulos se integran moviéndose hacia abajo a través de la jerarquía de control, comenzando con el módulo de control principal (programa principal). Los módulos subordinados al módulo de control principal se incorporan a la estructura ya sea en profundidad o primero en amplitud. El proceso de integración se realiza en una serie de cinco pasos: (22)

1. El módulo de control principal se utiliza como controlador de prueba y los apéndices se sustituyen por todos los componentes directamente subordinados al módulo de control principal.
2. Dependiendo del enfoque de integración seleccionado (es decir, primero en profundidad o primero en amplitud), los apéndices subordinados se reemplazan uno a la vez con componentes reales.
3. Las pruebas se realizan a medida que se integra cada componente.
4. Al completar cada conjunto de pruebas, se reemplaza otro apéndice con el componente real.
5. Se pueden realizar pruebas de regresión para garantizar que no se hayan introducido nuevos errores.

El proceso continúa desde el paso 2 hasta que se construye toda la estructura del programa.

Bottom-up (ascendente): Las pruebas de integración ascendente, como su nombre lo indica, comienzan la construcción y las pruebas con módulos atómicos (es decir, componentes en los niveles más bajos en la estructura del programa). Debido a que los componentes se integran de abajo hacia arriba, la funcionalidad proporcionada por los componentes subordinados a un nivel dado siempre está disponible y se elimina la necesidad de apéndices. Una estrategia de integración de abajo hacia arriba se puede implementar con los siguientes pasos: (22)

1. Los componentes de bajo nivel se combinan en grupos (a veces llamados construcciones) que realizan una sub-función de software específica.
2. Un controlador (un programa de control para pruebas) está escrito para coordinar el ingreso y salida de prueba.
3. Se prueba el conjunto o construcción.
4. Los controladores se eliminan y los grupos se combinan moviéndose hacia arriba en la estructura del programa.

En este caso se emplea un acercamiento práctico (a veces llamado Sandwichtesting) que utiliza Top-down para los niveles superiores de la estructura del programa y Bottom-up para los niveles inferiores. Combinando lo mejor de ambas técnicas. (22)

Detalles de la prueba:

La prueba se encuentra dividida en tres fases.

1. **Interacción de usuario:** En esta fase se revisan los datos de entrada y salida, el formato de presentación de los datos, el procesamiento de errores y la presentación de los mismos.
2. **Procesamiento de datos:** En esta fase se revisan la obtención de datos, manipulación del formato de los datos, determinación de condiciones de procesamiento, acciones requeridas como consecuencia de las condiciones encontradas.
3. **Comunicación:** En esta fase se revisan el traspaso de datos entre funciones y clases, la llamada a funciones, las rutas de comunicación entre clases y funciones.

En cada prueba se aplicaron los siguientes criterios:

- **Integridad de interfaz:** Son comprobadas las interfaces internas y externas de cada subsistema o funcionalidad que es integrada al módulo.
- **Validación funcional:** Se aplican pruebas diseñadas para descubrir errores funcionales.
- **Contenido de información:** Se aplican pruebas diseñadas para descubrir errores asociados con estructuras de datos locales o globales.
- **Desempeño:** Se aplican pruebas durante el diseño del software para verificar el desempeño del programa se encuentre entre los límites establecidos.

Ejemplo de prueba de integración.

CAPÍTULO 3

Una vez concluido el desarrollo de los módulos que forman parte de la Aplicación AKADEMOS para Android, de debe proceder a la realización de las pruebas de integración. En el marco de la investigación se aplica la prueba de integración al módulo “Control Docente”, obteniéndose los resultados que a continuación se muestran en cada fase.

Tabla 8.Caso de prueba de integración realizada al módulo Control Docente

Caso de prueba de integración del módulo: Reporte de los estudiantes de pregrado	
Sistemaal que se integra	AKADEMOS Android
Condiciones de ejecución	Tener acceso al sistema de autenticación y seguridad.
Descripción de la prueba	Comprobar que el módulo“Control Docente” aparezca correctamente en la pantalla principal del sistema, y que respete los niveles de accesos necesarios para poder interactuar con él.
Pasos de ejecución	El probador inicia sesión en el sistema con un usuario con nivel de acceso adecuado para poder interactuar con el módulo “Control Docente”. Se selecciona la opción Control Docente en el menú lateral de la pantalla principal del sistema. Comprobar que el sistema brinde el acceso correctamente al módulo “Control Docente”.
Resultados esperados	El componente “Control docente” brinda toda la información solicitada, visualiza los registros asociados al profesor y le permite registrar una asistencia de forma satisfactoria.
Evaluación	Prueba satisfactoria

3.4 Conclusiones del Capítulo 3.

En el presente capitulo se llevó a cabo el análisis del proceso de implementación y pruebas de la aplicación. La representación del diagrama de despliegue permitió identificar el comportamiento final y la interacción con las fuentes de información de la aplicación desarrollada. Es importante enfatizar en la integración con los servicios de la universidad, ya que se usaron tecnologías alternativas en el proceso de implementación. Las pruebas realizadas al software arrojaron resultados positivos, demostrando la calidad del producto final y la completa satisfacción de los requisitos planteados.

CONCLUSIONES

Conclusiones Generales.

Una vez culminado el proceso de desarrollo del sistema podemos arribar a las siguientes conclusiones de manera puntual:

1. El estudio de sistemas homólogos que permiten la gestión docente, posibilitó obtener una idea de los conceptos más importantes relacionados con la investigación.
2. La búsqueda de información referente a las tendencias actuales y tecnologías modernas permitió la selección del marco de trabajo y lenguaje de programación para aplicaciones como la desarrollada en el presente trabajo de diploma.
3. El levantamiento de requisitos llevado a cabo mediante la realización de Historias de Usuario permitió obtener de manera rápida y sencilla un correcto modelado del sistema obteniéndose un total de 15 Requisitos Funcionales y 9 Requisitos No Funcionales.
4. Los patrones de diseño seleccionados se adecuan totalmente a la arquitectura MVC propiciando gran facilidad en la implementación y mantenimiento del sistema.
5. Se logró desarrollar una aplicación Android que permite una correcta visualización y funcionamiento del módulo control docente de AKADEMOS en los dispositivos móviles.
6. Con las pruebas realizadas al software se garantizó la satisfacción total de los requisitos obtenidos durante el levantamiento, por lo que se cumple el objetivo de esta investigación.

Recomendaciones

Recomendaciones:

1. Extender el proyecto a cada uno de los restantes módulos del sistema de gestión académica de la universidad.
2. Hacerle pruebas a la aplicación en un entorno donde se tenga acceso a la plataforma de servicios de la UCI.
3. Desarrollar la aplicación para dispositivos móviles que utilizan sistema operativo iOS.

Referencias

1. Hernández Burbano, Angie Milena.*Desarrollo de uba Guía Metodológica para Diseño Web Adaptativo.* Quito, Ecuador : Pontificia Universidad Cátolica del Ecuadror, 2018.

2. ComputerHo. Qué es BYOD? ventajas e inconvenientes. [En línea] [Citado el: 16 de marzo de 2020.] https://computerhoy.com/noticias/moviles/que-es-byod-ventajas-e-inconvenientes-7250.

3. Departamento de Proyectos Europeos.*Diseñando el aula del futuro. Bring your own device (BYOD): una guía para directores y docentes .* Madrid : Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado (INTEF), 2016.

4. UCI. Misión | Universidad de las Ciencias Informáticas. *www.uci.cu.* [En línea] [Citado el: 16 de junio de 2019.] https://www.uci.cu/universidad/mision.

5. Molina Moreno, Pedro Juan.*Especificación de interfaz de usuario.* Valencia : Universidad de Valencia, 2013.

6. ISO.*IEC 9126.*

7. Gamma, Erich, Richard Helm Ralph Johnson and John Vlissides.*Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1995. ISBN 0-201-63361-2.

8. *¿Qué sabemos acerca de la eficacia de los patrones de diseño de software?* Budgen, C. Zhang y D. s.l. : IEEE Transactions on Software Engineering, 2012, Vol. 38. TSE.2011.79.

9. UCI. AKADEMOS. [En línea] [Citado el: 28 de enero de 2020.] https://akademos.uci.cu.

10. Educaciontrespuntocero. Educaciontrespuntocero.com. [En línea] [Citado el: 4 de febrero de 2020.] https://www.educaciontrespuntocero.com/recursos/plataformas-gestion-escolar/.

11. Partido Comunista de Cuba.*Actualización de los lineamientos de la política económica y social del Partido y la Revolución para el período 2016-2021.* La Habana : s.n., 2016.

12. Android Developer. *Android Developer.* [En línea] 2020. [Citado el: 18 de Junio de 2020.] developer.android.com.

13. desarrolloweb.com. desarrolloweb.com. [En línea] [Citado el: 10 de febrero de 2020.] https://desarrolloweb.com/home/java.

14. docs.gradle.org. docs.gradle.org. [En línea] [Citado el: 12 de febrero de 2020.] https://docs.gradle.org/current/userguide/userguide.html.

15. developer.mozilla.org. developer.mozilla.org. [En línea] [Citado el: 16 de febrero de 2020.] https://developer.mozilla.org/es/docs/Web/XML/Introducci%C3%B3n_a_XML.

16. ecured.cu. ecured.cu. [En línea] [Citado el: 17 de febrero de 2020.] https://www.ecured.cu/PgAdmin3#:~:text=Es%20una%20aplicaci%C3%B3n%20de%20dise%C3%B1o,sobre%20casi%20todas%20las%20plataformas.&text=La%20interfaz%20gr%C3%A1fica%20es%20compatible,PostgreSQL%20y%20facilita%20la%20administraci%C3%B3n..

Referencias

17. www.campusmvp.es. www.campusmvp.es. [En línea] [Citado el: 18 de marzo de 2020.] https://www.campusmvp.es/recursos/post/Herramientas-de-prototipado-de-aplicaciones-Web.aspx.

18. Cuervo, Víctor. www.arquitectoit.com. [En línea] [Citado el: 23 de febrero de 2020.] http://www.arquitectoit.com/postman/que-es-postman/.

19. https://nodejs.org/es/. nodejs.org. [En línea] [Citado el: 25 de marzo de febrero.] https://nodejs.org/es/..

20. www.json.org. https://www.json.org/json-es.html. *https://www.json.org/json-es.html.* [En línea] [Citado el: 14 de junio de 2020.] https://www.json.org/json-es.html.

21. jwt.io. Introduction to JSON Web Tokens. [En línea] [Citado el: 25 de junio de 2020.] https://jwt.io/introduction/.

22. Pressman, Roger S. *Software Engineering. A Practitioner's Approach.* New York : McGraw-Hill Education, 2015. 8va edición.

23. Tamara, Sánchez Rodríguez. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015 .

24. proyectosagiles.org. Qué es SCRUM – Proyectos Ágiles. [En línea] [Citado el: 14 de noviembre de 2019.] https://proyectosagiles.org/que-es-scrum.

25. pmoinformatica.com. Requerimientos no funcionales: Ejemplos. *pmoinformatica.com.* [En línea] [Citado el: 17 de Junio de 2020.] pmoinformatica.com.

26. ionos.es. www.ionos.es. [En línea] 17 de octure de 2018. [Citado el: 21 de junio de 2020.] https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/como-crear-una-app-disenar-apps-moviles/.

27. Ledesma, E. Certificado en Fundamentos de Scrum SFCTM GRATIS. [En línea] [Citado el: 17 de mayo de 2020.] https://www.tenstep.ec/portal/articulos-boletin-tenstep/41-scrum/253-scrum-como-escribir-historias-de-usuarios-sin-morir-en-el-intento.

28. *IBM notice: The page you requested cannot be displayed.* notice, IBM. 2016.

29. Maniakhitoccor. Los 10 patrones comunes de arquitectura de software. [En línea] 17 de mayo de 2020. https://medium.com/@maniakhitoccori/los-10-patrones-comunes-de-arquitectura-de-software-d8b9047edf0b.

30. si.ua.es. Modelo vista controlador (MVC). [En línea] [Citado el: 17 de mayo de 2020.] https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html..

31. Segundo M. A., Hernan. José A. A. ,Joaquin. *Diseño e Implementación de un Servicio Web (Web Services) Para la Búsqueda de Hoteles en la Zona Costanera del Departamento de Córdoba.* 2015.

32. Craig, Larman. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Chicago : s.n., 2000.

33. Departamento de Lenguajes y Sistemas Informáticos. *Patrones de Asignación de Responsabilidades (GRASP).* Sevilla : Universidad de Sevilla, Escuela Técnica Superior de Ingeniería Informática.

34. Pérez Mariñán, P. *Patrones de Diseño (Design Patterns).* 2003.

Referencias

35. *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web.* C. A. Guerrero, J. M. Suárez, y L. E. Gutiérrez. 3, s.l. : Inf. Tecnológica, 2013, Vol. 24.

36. campusMVP. *Diseñando una base de datos en el modelo relaciona.* [En línea] [Citado el: 17 de mayo de 2020.] : https://www.campusmvp.es/recursos/post/disenando-una-base-de-datos-en-el-modelo-relacional.aspx.

37. *Introducción a las Pruebas de Sistemas de Información.* Toledo Rodríguez, Federico . 2014.

38. *Un Sondeo sobre la prática actual de pruebas de software en Espanã.* Fernández Sanz, Luis. Madrid : Reicis Revista Española de Innovación, Vol. 002, págs. 43-54.

39. Pressman, Roger S. *Ingeniería del software.Un enfoque práctico.* New York : McGraw-Hill, 2010.

40. Joskowicz, José. Reglas y Practicas en eXtreme Programming. [En línea] [Citado el: 16 de mayo de 2020.] https://www.academia.edu/download/31398587/xp_-_jose_joskowicz.pdf.

41. developer.android.com. developer.android.com. [En línea] 1 de junio de 2020. [Citado el: 14 de julio de 2020.] https://developer.android.com/training/multiscreen/screendensities?hl=es-419.